Using Machine learning for Feature Detection in Transmission Electron Microscopy

by

Changjian Cao

B.S., East China University of Science and Technology, 2015

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Changjian Cao

It was defended on

March 26th 2019

and approved by

Tevis Jacobs, Ph.D., Assistant Professor,

Department of Mechanical Engineering and Materials Science

Hessam Babaee, Ph.D., Assistant Professor,

Department of Mechanical Engineering and Materials Science

Murat Akcakaya, Ph.D., Assistant Professor,

Department of Electrical and Computer Engineering

Thesis Advisor: Tevis Jacobs, Ph.D., Assistant Professor,

Department of Mechanical Engineering and Materials Science

Using Machine learning for Feature Detection in Transmission Electron Microscopy

Changjian Cao, M.S.

University of Pittsburgh, 2019

In situ testing performed in a transmission electron microscope (TEM) represents an important technique for materials analysis. However, it produces a large amount of data in the form of images and video, which cannot typically be analyzed using traditional image-analysis algorithms. Therefore, a machine-learning approach is proposed to detect the shape of a body by recognizing and locating the border of the material. This supervised-learning algorithm is applied and a convolutional neural network is built to rapidly label all pixels. This network explores the relationship between small sub-images cropped from original images and their corresponding labels, and then it predicts the label when given new sub-images, thus generating a segmented image. In this project, the performance was assessed based on specificity, sensitivity, and accuracy of results. The overall accuracy of the present model is over 90%; however, the precision and recall rate are low due to high false-positive detection. This research suggests key factors for improving future machine-learning algorithms for TEM image analysis.

Keywords: Transmission electron microscopy, Convolutional neural networks, Supervised learning, Image processing.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

As the development of sciences and technology, machines have taken human's role in many fields, like in the manufacture, construction, and transportation. Even though machines are seen everywhere, the task they work on is either designed by human or cooperated with workers, so one idea raises for scientists, that is, can machines learn to complete a task by themselves? Inspired by this thought, artificial intelligence and machine learning have been proposed. To make this thought into reality, scientists propose and create many new hypothesis and tools, and the neural network model is invented at that time. As a machine learning algorithm, neural networks refer to networks which are able to learn knowledge from given data and then they tackle new problems based on what it has learned.

Due to the development of machine learning technologies in recent years, neural networks have proved their effective performance in some fields. One notable example is the handwritten digit recognition[1] with neural networks, this neural network learn the knowledge from given images which contains digital numbers, and then it outputs a prediction for the digit number when given a new image. However, even the neural networks take a big advantage in those fields, some critical drawbacks start to limit their applications, and their ability is also doubted by humans. Therefore, new and more powerful technology is proposed, that is, Convolutional neural networks(CNN). Unlike neural networks that destroy the spatial information of the detected objects by stretching images into one-column values, convolutional neural networks operate on the original images. Working on the original images leads to that the convolutional neural networks are more robust to the noise and variances, which means that this kind of networks is able to figure out the objects on the different position of images, even that items that they try to recognize are rotated, flipped or their size is changed.

As the very critical convolutional neural network, the Alexnet model was created by Alex Krizhevsky et al.[2]. Due to the invention of Alexnet, the precision in the competition to classify the LSVRC-2010 ImageNet dataset increased sharply, which boosts the development of convolutional neural networks. Since then, more new convolutional neural networks, like VGG[3], GoogleNet[4] and Resnet[5], have been created. And gradually, convolutional neural

networks become widely used in many fields, like image classification, image segmentation, and object recognition. Even there are many traditional image processing methods to process images and video in those fields, CNN still shows a huge advantage over these methods. Comparing with the traditional methods, CNN is more precise in recognizing objects, at the same time, besides recognition, they can figure out the position where the object locates. Recently, many projects and researches are conducted on the convolutional neural networks. Cha KH et al.[6] developed a deep-learning convolutional neural network to segment the bladder in CT urography, and the network is capable of distinguishing the regions inside and outside of bladder and thus generating a clear contour segmentation. S. N. Sangeethaa et al.[7] also proposed CNN to segment the blood vessel to diagnose the diabetic retinopathy, the neural network which they created run on the pre-processed images, and they got an over 95% accuracy for disease diagnosis in their research. Bijen Khagi et al.[8] constructed a deep neural network to segment the MRI images, they employed a pixel-label-based method to generate a label for each pixel and combine the all labeled pixel to build a good segment map.

Considering the prospect of CNN models in computer vision, it's possible to apply this method to analyze materials in transmission electrons microscopy images. To process the transmission electron microscopy images and detect the border of materials, several traditional methods are widely applied, including manually labeling and image processing. When compared to the machine learning method, manually labeling is time-consuming and the reliability is solely determined by raters, image processing, though easily implemented, is greatly undermined by noise. However, machine learning method requires less human effort and behaves more robustly against noise. These advantages result from that only a small set of human-labeled data is involved prior to model training, and machine learning method more focuses on semantic information of images.

In this thesis, we apply the supervised learning algorithm and build up a convolutional neural network to detect the border of materials in images. In chapter 2, we describe the application of TEM in materials' structure analysis, supervised learning algorithm, and some information about convolutional neural networks and image segmentation. In chapter 3, we display the methods we adopt in this project, including preliminary data processing,

convolutional neural network training, and performance evaluation. In chapter 4, we display some results we get from the CNN model and analyze those results. In chaptr 5, we point up the limits in this task and propose some improvements for our future work.

2.0 Background

2.1 Transmission Electrons Microscopy

As a powerful technique to detect the structure of specimens, the transmission electrons microscopy(TEM)[9] captures the features by applying an accelerated beam of electrons to interact with specimens. Benefit from this special method to form images, TEM is widely used in different fields, including nanotechnology, medical, biological and material research. Generally, compared with other light microscopes, transmission electron microscopes take images at a powerful magnification and those images are in fine details. With transmission electron microscopes, it can extract information about surface features, including shape, size, and structure. The figure1 gives an example of a transmission electrons microscope.

Figure 1: Transmission electrons microscope[10]

2.2 Supervised Learning and Unsupervised Learning

Supervised learning and unsupervised learning, which are two major fields in machine learning, are intensively applied in classification and regression problems. The obvious difference between these two methods is whether the training data for models is labeled or not. For the supervised learning algorithm, it is originally designed to train a model by the instance with labels, and then to predict the test samples' label in term of predictor features[11]. While in most cases, manually labeling the objects are expensive, and then the unsupervised learning algorithm is proposed. For the unsupervised learning algorithm, data tend to cluster in different groups based on their similarity in general, but the result is easily affected by the noise. Compared with the unsupervised learning, the supervised learning algorithm is more robust and the result is more precise. Both methods include many different algorithms, decision Tree[12], logistic regression, support vector machine[13] and neural networks are main algorithms in supervised learning, and k nearest neighbors, expectation-maximization algorithm are typical algorithm as the unsupervised learning method.

Support virtual machine, as an example of supervised learning algorithms, is proved very powerful in classification problems. In a support vector machine model, when given a set of training samples, the support vector machine algorithm will map those data into a high-dimension space if data can't be separated in a lower plane. At the same time, a function will be generated to classify data in that high-dimension space. Generally, data that belong to different classes have distinctive characteristics and features, the support vector machine algorithm will identify their special features and find functions to distinguish objects based on their different features or patterns. Normally, the number of functions that can be applied to classify data might be infinite, but why support vector machine is always the top choice to generate the function is that this method tries to find a function which widens the distance between the samples and the trajectory of the function as much as possible. The figure2 gives one example of using support vector machine method to implement classifications. The red ball and the blue ball belong to two different class, and they can be separated into two classes with the red line and the blue line. The distance between the samples and the red line is bigger than that of green line, which means the discrepancy from different classes is

more obvious when applying this function to these data. Based on this method, It is less possible to classify the data by mistakes. Thus, the support vector machine algorithm is frequently used for classification in the supervised learning field.

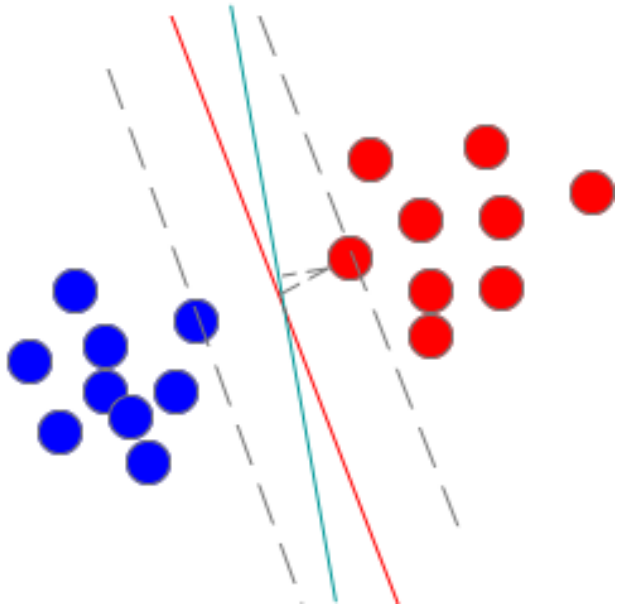

Figure 2: Support vector machine

On the other hand, one typical unsupervised learning algorithm is k-Nearest Neighbor. As the name indicates, the K-nearest neighbor algorithm refers to one algorithm that finds K data which are closest to the central data and lets the unknown point belong to the class label which the major data in these k data takes. The figure3 shows how K-nearest neighbor algorithm works. To determine the class of red circle, we need to first choose the k values, if k is set to be 3, then we calculate the distance between the red circle and its three closest samples. There are two green rectangles and one triangle in these three samples, so the red circle is assigned to the major samples' class, that is the class which green rectangles belong to. Similarly, if the k value is chosen as 7, then there are four blue triangles in the seven samples, so the red circle has the same class label as the blue triangle. The K-nearest neighbor method is a lazy algorithm, it doesn't need to find the parameters and construct discriminant functions, all it needs is to calculate the distance between the test sample and

the samples around it, then predict the class of test sample based on this distance and the k value. K-nearest neighbor algorithm is also a easy-to-implement algorithm because of the simple calculations.

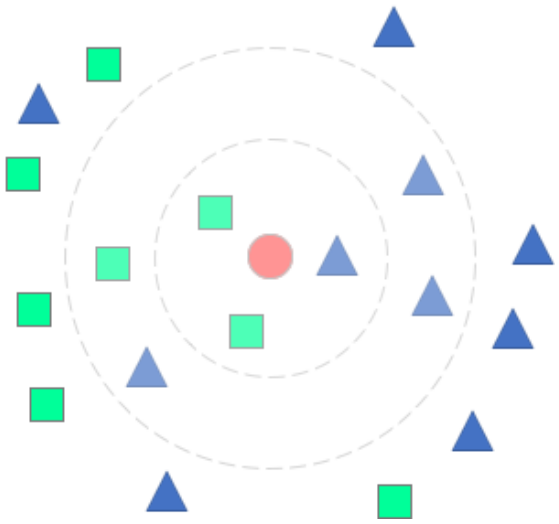

Figure 3: K-Nearest neighbor algorithm

2.3 Convolutional Neural Networks

As one typical supervised learning algorithm, convolutional neural networks take good advantages than other supervised learning algorithms, especially in deep learning. Developed from the cat's visual cortex experiment conducted by Hubel and Wiesel[14], convolutional neural networks have a similar architecture as the human visual system and thus getting the capability of perceiving and identifying objects. As figure4 shows, convolutional neural networks consist of several components including layers, activation and output functions. Main structures are constructed by convolution layers, pooling layers, and fully connected layers, and the activation functions include sigmoid, Relu, and tanh et al., for the output functions, softmax and sigmoid are frequently used. Based on the diagram, the images are first feed into the convolution layers of the model, and the following layers operate on the output map

from the previous layers, after the inputs go through all convolution layers, the model will extend the output map and output a probability calculated by the output functions, this probability shows the class which the data belongs to.

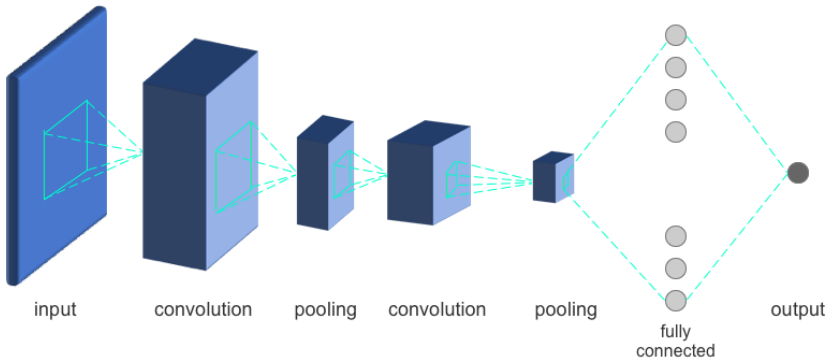

Figure 4: Convolutional neural networks architecture

Convolution layers, as the major component of the networks, play a critical role in generating the relation between inputs and output. Normally, the convolution layers consist of several small 3-D filters, each 3-D filter have the same depth as that of inputs, and values of parameters in filters are unknown. At the same time, the depth of output from convolution layers solely depends on the number of filters. In most deep convolutional neural networks, the filter size in convolution layers is very small, and 1x1, 3x3 and 5x5 size are generally chosen. When given inputs, each small 3-D filter will slide on the input maps and convolve with values in the same patch that it covers. The sum of these results make up one value on output maps, then these 3-D filters move to next position and generate other output. The step that filters move in each time depends on one chosen parameter, which is "stride". The bigger the stride is, the wider steps filters move, and the size of outpur maps is also smaller. Sometimes, besides convolving directly with filters, inputs can also be padded first and then feed into convolution layers. By padding, the size of inputs become larger, when the filters and stride are same, the output maps will also be larger. In each convolution layer, the parameter convolves with the input maps and output new maps, after being operated with nonlinear functions or other methods, these new maps are as inputs to feed into the

following layers. In convolutional neural networks, these maps are also called feature maps. The size of feature maps depends on the filter size, stride and padding size, it's defined as equation2.1. In the equation, "i" denotes the size of inputs, "k" denotes the size of filters, "p" denotes the padding size, and "s" denotes the stride.

$$o = \frac{(i - 2 * k + 2 * p)}{s} + 1 \tag{2.1}$$

The relationship between inputs and feature maps is also determined by nonlinear functions, which are known as "activation function". By introducing nonlinear functions, the whole network becomes nonlinear, and then it can derive kinds of functions to classify data. Actually, The underlying assumption that CNN is good at classifying objects is that networks can approximate universal functions, so it can always construct a proper function to separate different data. As the critical part of whole networks, activation functions are commonly chosen from several nonlinear functions, including sigmoid, Relu and tanh et al.. Relu has been more intensively used in deep convolutional neural networks recently, and it is defined by equation2.2, this function is designed to output non-negative numbers no matter how the given inputs are positive or not. As it shows, when the inputs are negative, Relu function turns them to zero, otherwise, it keeps the value. This phenomenon can also be found in the neural system, in figure5, the axon receives the signals from the previous neurons and selectively transfer information to the following dendrites of other neurons[15].

$$f(x) = max(x, 0) \tag{2.2}$$

Another important architecture for CNN is pooling layers, and they are viewed as a critical tool to shrink feature maps and decrease the computation complexity in machine learning. Pooling layers extract value from each small patch on inputs and generate output only with these chosen value. Max pooling and average pooling are two Commonly used pooling types, the figure6 shows the basic structure of Max pooling. In Max pooling layers, the maximum of each 2x2 patch will be chosen as an output, at the same time, the size of the output maps decreases to one-fourth of original inputs. Even though the feature maps size shrinks, it is enough for a network to construct functions and classify objects. As we know, images that drop values in a small local region will still show the same object but

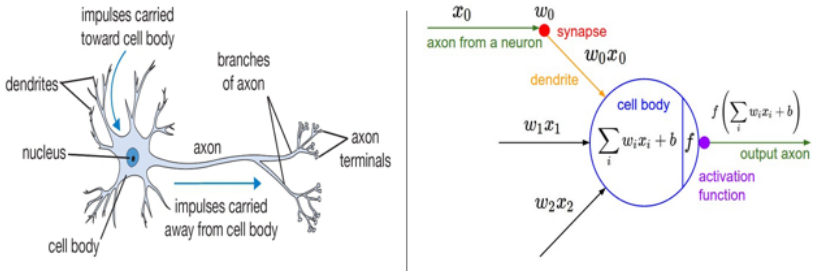

Figure 5: Neural system and activation function[15]

with lower resolution, which is also names as transfer invariance. In classification problems, all we care about is whether the detected object is in this image or not, the polling layers only decrease the size of the object but the exist of objects doesn't change. By introducing pooling layers, the size of feature maps reduces, compared with networks that only apply convolution layer, fewer convolution layers are required to obtain same size output maps.

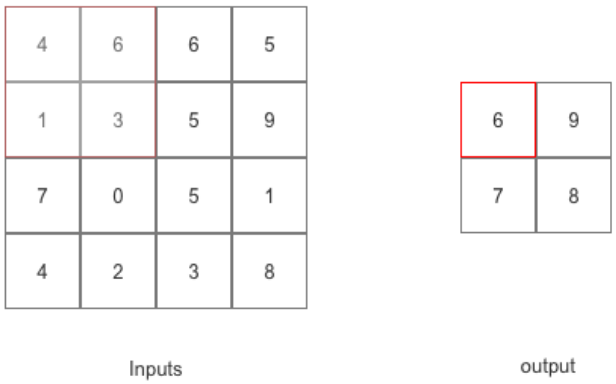

Figure 6: Pooling layers

Still, fully connected layers are also regarded as one of the fundamental components, and they're always introduced as the last several layers of networks. Fully connected layers are added for stretching the 2-D feature maps into a signal vector, then this vector is convoluted with fixed size 1-D filters and output results. However, fully connected layers cause several serious issues for networks. One is that the size of inputs is constrained by fully connected layers. when constructing networks, the size of 1-D filters in fully connected layers is already chosen, and filters with fixed size only accept one special size inputs, which limits the use of networks in variable inputs projects. Also, fully connected layers contain huge weights, and these weights take over half of total weights in networks. And a huge number of weights will cause a simple issue, that is, it takes lots of times to train these weights. At the same time, fully connect layers also easily result in overfitting. Overfitting is always a serious issue that scientists try to prevent in machine learning, it means that the network performs well in training data but it works badly in testing data.

To handle this problem, two methods are applied. One is discarding the fully connected layers and replacing them with global average pooling layer. Instead of reshaping feature maps to one vector, global average pooling layers just use filters with the same size as feature maps and generate one result for every single map, in the other word, if the feature maps as inputs to global average pooling layers are 8x8 and then the filter will be 8x8. Another is introducing dropout[16] to fully connected layers. The figure7 describes how dropout works. In the fully connected layers without dropout, all of the units are connected whenever in training or testing, while in the fully connected layers with dropout, the networks will randomly drop out some units during training but include all units when testing data. Dropping some units results in different units participating in training, thus generating different networks to learn from data. When testing data, all the units are contributed to the output in the fully connected layers. So the output is actually the average result of the different networks generated in training, which prevents a network from suffering from the uncertainty from a signal model and thus decreasing the possibility of overfitting.

The final goal for convolutional neural networks is to classify objects, then possibilities that determine which class object belongs to are needed. To give a possibility of objects, one output function is introduced as the last component to the networks, and the most

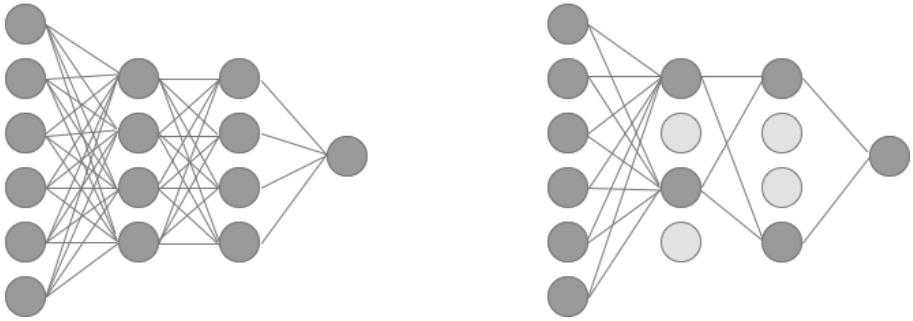

Figure 7: Fully connected layers: No dropout(left); Dropout(right)

frequently applied functions are sigmoid and Softmax. The sigmoid function is mainly used in binary classes classification since we only need one possibility to determine if the object belongs to this class or not. The equation2.3 gives the definition of the sigmoid function, it outputs a value in the range between 0 and 1. If the probability is higher than 0.5, then the object belongs to this class, otherwise, it belongs to other class. Softmax is always applied in multi-classes problems. In multi-classes classification projects, only one probability is not enough to tell that object's class, so Softmax function is introduced. As the equation2.4 illustrates, given a set of inputs, softmax function will output results corresponding to the percentage of each value in total sum, and this percentage is also viewed as the probability. What we need to do is to figure out the maximum of these probabilities, if "x_i" is the highest, then this object is predicted as the ith class.

$$f(x) = \frac{1.0}{1 + \exp(-x)} \tag{2.3}$$

$$p(x_k) = \frac{\exp(x_k)}{\sum_{k=1}^{n} \exp(x_k)} \tag{2.4}$$

When the convolutional neural network is already built, then we have to train that network. As the last and most important step in machine learning, training networks refer to optimizing the arbitrary parameters in networks and thus making the network to fit

training samples well. To train model, we have to feed training samples and their given labels, and then compare these actual labels and predicted labels obtained by the network. So, a function that measures the difference between the actual label and predicted label is created, and this function is known as a cost function. Mean square error and cross entropy are two commonly used cost functions, the equation2.5 and equation2.6 show the definition of mean square error and cross-entropy separately. In the equation, "y_i" denotes the predicted label, "$\hat{y_i}$" denotes the actual label, and "n" denotes the sample size. When cost function is chosen, then weights in different layers will be updated based on its result. In deep learning, constrained by their deep structure and huge weights, it's not easy to directly find the optimal values for neural networks. Thus, to handle these problems, the gradient descent algorithm and chain rule method are adopted. On the one hand, the gradient descend method provides a decent way to search optimal values for big weights, especially there are nearly millions of weights For deep networks. And as it refers to before, the cost function is convex, which means that weights in networks will always reach their optimal value with this method. On the other hand, chain rule helps to update weights in the different layers. After obtaining the derivative of the cost function with respect to weights in the last layer, networks are able to transfer these values to the beginning through chain rule and update all weights at the same time. Overall, the updating function is defined by equation2.7, in the equation, "$\hat{w}$" denotes the new weights, "w" denotes the old weights, "$\frac{\partial cost}{\partial w}$" denotes the derivative of cost with respect to weights, and "α" denotes the moving speed in the gradient direction.

$$cost = -\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2.5}$$

$$cost = -\sum_{i=1}^{n}(\hat{y_i} * \log y_i + (1 - \hat{y_i}) * \log(1 - y_i)) \tag{2.6}$$

$$\hat{w} = w - \alpha * \frac{\partial cost}{\partial w} \tag{2.7}$$

2.4 Semantic Segmentation

Image classification, images segmentation, and object recognition are three major applications in machine learning. Unlike images classification, images segmentation needs to not only classify each object but also figure out its location. To locate objects, two kinds of networks with different structure are proposed. one is like V-shape networks and another one is the networks with dilated structures.

A typical V-shape network which is the fully convolutional network was first introduced into the field of images segmentation by J.[17]. The figure8 display its basic architecture. The fully convolutional network is mainly constructed by two parts, the first part is a convolutional neural network, the second part is a transposed convolution structure. When inputs are feed into networks, they will be compressed by the convolution layers and turn out to be smaller maps, and then this network reconstruct a map with the following transposed convolution[17] layers, the final output in this network is the segmented image with the same size as inputs. Also, In the first half part, the fully convolutional network discards all the fully connected layers and replace them with convolution layers, so the spatial information of pixel will be kept in the whole model. After obtaining the output maps from the convolutional part, transposed convolution layers expand that small maps and combine them with a same-size output in the first part, finally a segmented image with the same size as inputs are generated directly. The assumption about reconstructing the image based on the predicted output from the first half part is that each intermediate feature maps contain the information from inputs in previous maps, so it's possible to generate an image from a relatively small map gradually. The figure9 shows the operation of convolution in a local region, observing the figure, we know that even the size of feature maps shrinks because of convolution layers, the value on output maps still contains the information from that region in the previous layers, which provides an opportunity to reconstruct the original maps. In the figure10, we also illustrate how transposed convolution works. As we see, the input map is 4x4 and kernel is 3x3. By transposed convolution, each value on that small input map will multiply with this kernel and give a 3x3 output map, and the adjacent values on inputs will generate overlapped maps, then values in the overlapped area are summed to derive

new results. In general, the size of the output maps is 6x6, which is bigger than the inputs. Therefore, with fully convolutional structure and transposed convolution layers, the fully convolutional network creates an end-to-end method for image segmentation.

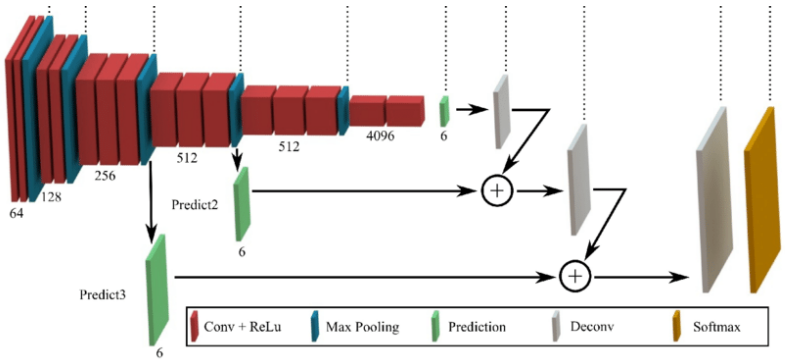

Figure 8: Fully convolutional neural networks architecture[18]

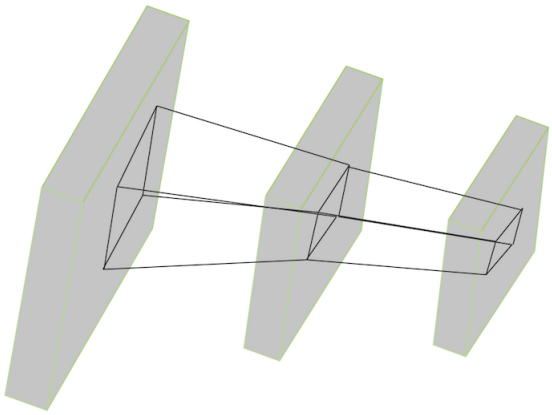

Figure 9: Information transfer through different convolution layers

As another application in the field of images segmentation, dilated convolution structure was proposed by Yu, F.[20] to segment images in a totally different way. In a typical convolutional neural network, kernel slides on input maps and convolves with pixel values

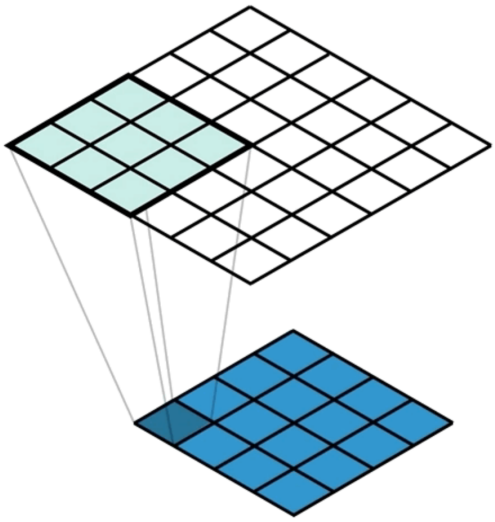

Figure 10: Transposed convolution[19]

continuously. While the dilated structures apply a hole algorithm when convolving, which means that kernel actually works like be adding some holes between rows and columns when convolving with input maps. The figure11 shows how a basic dilated convolution works. In the figure, the input map is 5x5 and kernel is 3x3. As it mentions above, in a typical convolution layer, this small kernel will get convolution with the patch it covers when moving on the input maps, so the value in this small patch is actually continuous. And for a 5x5 input maps, the final output it generates is a 3x3 feature map. While in dilated convolution layers, only a 1x1 feature map is derived with the same filters. Comparing with convolution layers, The dilated convolution layer works in a totally different way. When getting convolved with inputs, the 3x3 kernel in dilated convolution layers will be expended to 5x5 with zeros filled between rows and columns, and then this kernel slides on input maps and generates output maps. In other terms, the filter only convolves the values within constant distance on input maps, and this distance is controlled by a setting parameter. One advantage to convolutional neural networks with dilated convolution structure is that each value on output map

will contain information in a wider region, which helps network combine more information to explore relations between inputs and their labels. This method also makes sense, especially in the computer vision field. The underlying assumption that combining information within a wider region helps network get higher accuracy for detecting objects is that networks might be limited to learn features in the small area, as we know, the changes from pixels in a small continuous region are very small, this network might not tell the difference among them. However, by choosing varied distances, the network with dilated convolution structure is able to combine result from different kinds of regions, and then it will generate a more precise function to classify different objects. Networks with dilated convolution structure are also able to maintain the size of feature map by padding input maps, thus, by adding padding and applying dilated convolution structure, convolutional neural networks will output segmented images directly.

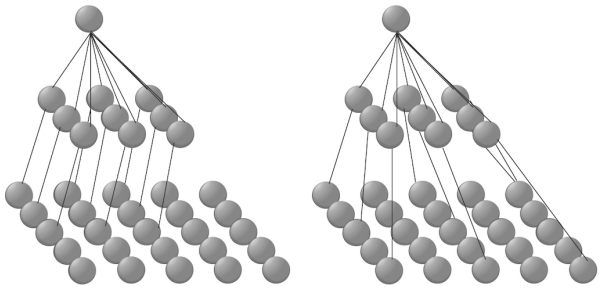

Figure 11: Convolution architecture: convolution(left); atrous convolution(right)

3.0 Methodology

In this chapter, we will describe the details of the methods we apply in this project, the whole procedure is given by figure12. Based on the diagram, the process is composed of three parts, including data preprocessing, model testing and performance evaluation. In data preprocessing, we collect TEM images in which the border points are manually labeled, then an interpolation method is employed to link the labeled points and thus achieve a continuous border. After obtaining the images and interpolated border, we crop the original images into small patches of sub-images, the label of sub-images depends on its central point, and then those sub-images and their corresponding labels are divided into training set and testing set. When the CNN model is properly trained based on a training set, we will test the pre-trained model. First, we will apply this model on testing images and thus acquire the predicted labels, with those labels, we generate the segmented images which are in the same shape of original images. Based on the result, we will consider whether to implement post-processing method or not. In the end, the performance of this model is assessed by comparing the ground-truth with the predicted segmented images. More details can be found as follows. In the section3.1, we refer to basic information about our collected images. In section3.2, we describe the methods that we use in processing images and augmenting data. In section3.3, we display the architecture of our network and explain the structure of the model. And in section3.4, we show the basic setting for parameters and main methods we apply in the training process. Finally, in section3.5, some measurements are given to assess the performance of this network.

3.1 Data Collection

To get data for the training set and testing set, we collected forty-four high-resolution TEM images, in which the materials and background are concluded. For each TEM image, the size is 2048x2048 pixels.

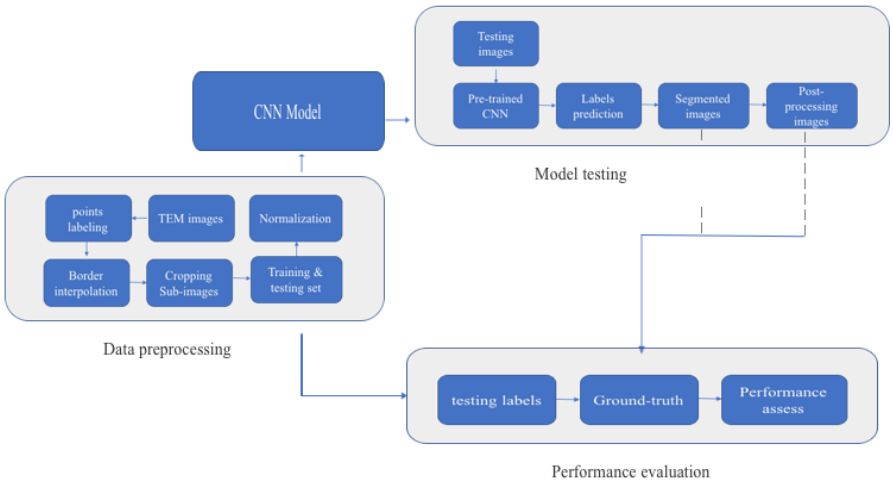

Figure 12: Procedures for border detection

In this task, considering the fact that several images have a similar structure of materials' border, we will assume only three main different types of borders in the whole task, and the figure13 shows the three main types of border images. In these images, we can easily tell that there are two different parts in whole images. In those images, the area in a deeper color is the projection of materials, while the lighter area is occupied by the background, the border of material locates at the interface of two different areas. To balance the ratio among different types of borders, only thirty-two border images are chosen. Those selected images are divided into a training set and testing set. In the training set, Twenty-nine border images are picked up, and three images are from type 1, sixteen images are from type 2, and ten images are from type 3. In testing data, we select three images in which all types of the border are contained.

According to the requirement of the supervised learning algorithm, we need to not only collect border images but also manually label the border points in these images. In this whole task, only two class labels are involved. When manually labeling, We pick up points which are obviously located at the border of materials and their positions are recorded. Overall, in every single image, about two hundred and fifty border points are labeled.

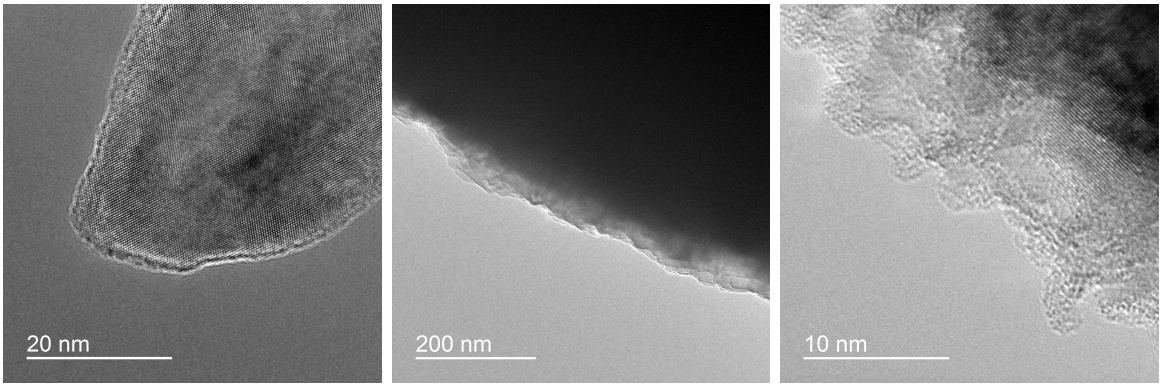

Figure 13: TEM images for different types of border

3.2 Data Processing

3.2.1 Point Interpolation and Ground-truth Generation

To train the model, directly applying these collected data is not proper. One obvious issue is that these human labeled border points, which is supposed to be as the desired output, are discrete. While the real border of materials generally is a continuous line, this discrete labeled points cause missing information about the border between each two adjacent border points. Thus, it's hard for the CNN model to figure out the connections between images and their real labels because of the lack of information about the border. As a result, the predicted border this network generates won't match the real border of materials. In order to address the issue from the discrete border, we have to link these isolated points first. To find the missing border points, an interpolation method is applied. With bilinear interpolation algorithm, the original data are subsequently interpolated, producing a continuous border. The point which locates on this new border will be assigned to be a border point. The figure14 shows the original human labeled border points and a continuous border line generated with interpolation. After interpolation, the number of new border points is nearly ten times more than the original one.

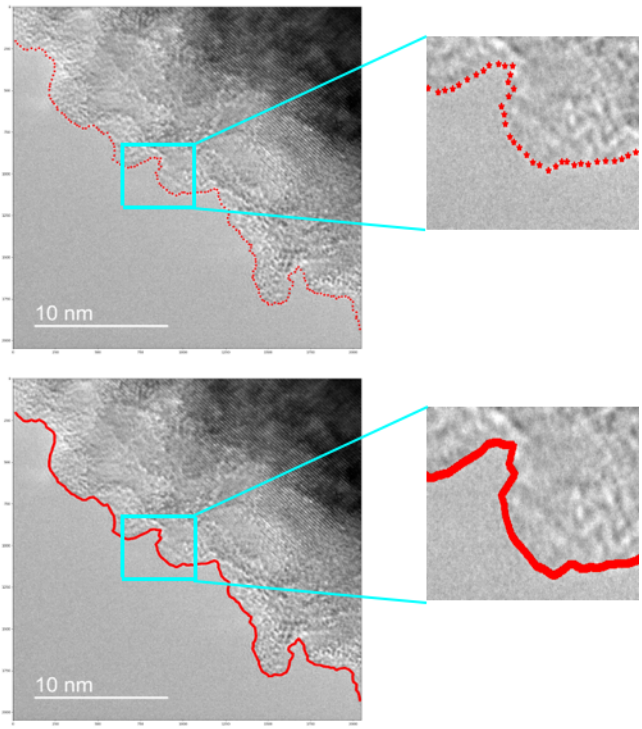

Figure 14: Material border: without interpolation(top), with interpolation(bottom)

After interpolating these discrete border points, corresponding ground-truth images are created. As the final desired output, those ground-truth images are a construct based on the points' labels and their positions. In this project, we only need to classify points on the border from those in the background, so the value of one point will be 1 in ground-truth images if this point locates on the border of materials in original images, otherwise, that value will be assigned to be 0. The figure15 shows the original image and the corresponding ground-truth image. In the ground-truth image, the white line indicates the border and the width of this border is only 1 pixel, all the black area is background.

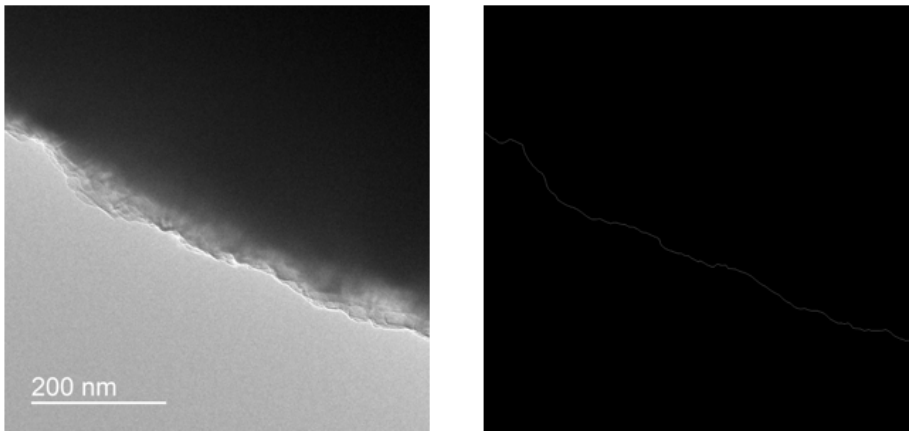

Figure 15: Image and ground-truth: original image(left), ground-truth(right)

3.2.2 Sub-images Cropping and Data Augmentation

For a CNN model, the size of inputs highly affects the computation complexity of the network. Recalling that the size of original images is 2048x2048, directly choosing these images as inputs will require a deeper structure and heavy parameters in a network, thus causing a high computation complexity. Therefore, sub-images with small size will be used as input for our network. Based on the method, we first crop the original images to lots of small pieces of sub-images. The labels of those sub-images solely depend on their central point, the sub-images' labels are assumed as border if the central point of this sub-image locates on the border, otherwise, the label will be assigned as background.

Before cropping images, the size of the sub-images is needed to be set. The figure16 shows the sub-images in different sizes. Observing these sub-images, we find that the sub-images with 65x65 size is too small to distinguish the points in the background from on border, which means extracting the border information is hard with this shape. For the 257x257 size sub-images, in which the trajectory of the border is very clear, but this big-size map causes a higher computation complexity. However, a clear border can also be found

in sub-images with 129x129 size. Moreover, choosing this relatively small size sub-images as inputs requires fewer computations. After determining the size of sub-images, we must consider the ratio of different types of borders. To balance sub-images which are generated from different types of borders, we get over 250000 sub-images as the whole training set. In these training data, 72000 sub-images are obtained from type 1, 108000 sub-images are from type 2, and 80000 images are from type 3, the ratio among these sub-images is 2:3:2.

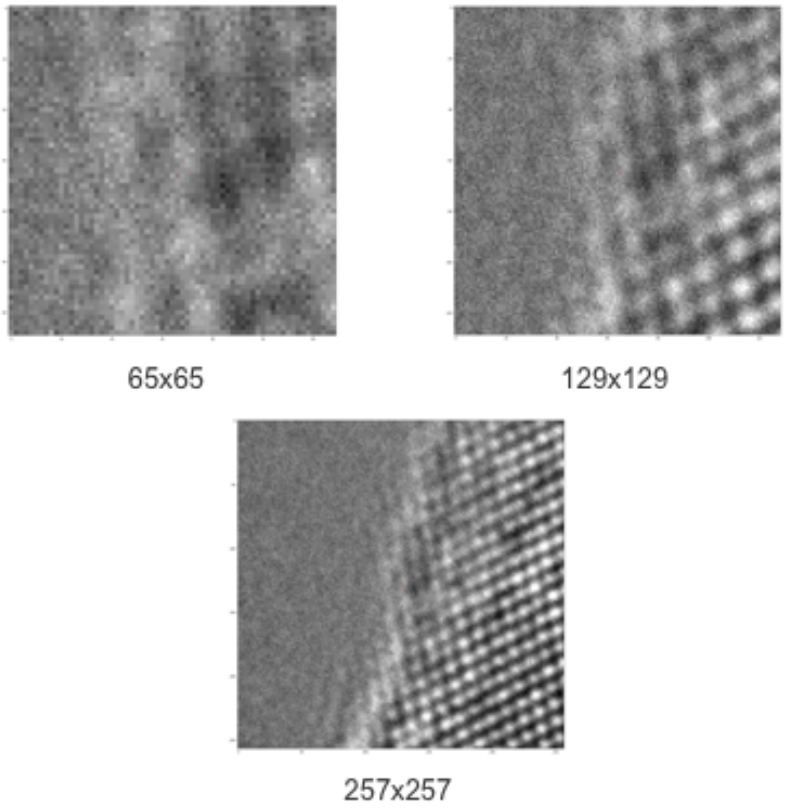

Figure 16: Sub-images with variable sizes

It's also worth mentioning that border points are rare when compared with the points in the background, and the imbalance between border points and background will greatly affect the performance of the network. With huge sub-images cropped from the background as a training set, the model will learn more information about features of background and thus acquire little knowledge about border points. To address this problem, a data augment method is implemented to increase the number of sub-images from the border. As it mentions

above, the border images of type 1 are rare, accordingly, the number of border points is also small, so we rotate the sub-images after cropping type 1 original images. while the number of border points in other types of border images is enough to form the training set, then we only crop these border images. After getting the augmented data from different borders images, we have to balance the ratio between points on the border and in the background. In this project, the ration is set to be 30:70. The figure17 shows points distribution in one image. In that figure, the yellow points denote data picked up in the projection of materials, red points denotes data picked up around the border, and blue points are picked up in the background, the green line shows the material's border. Overall, we get 8000 sub-images in every single image, in which 30% of the sub-images belongs to border class, and the remaining belongs to background class. And in all sub-images cropped from the background area, 20% data is chosen from the area around the border, 20% is from the outside of background, and the rest is from the projection of material.

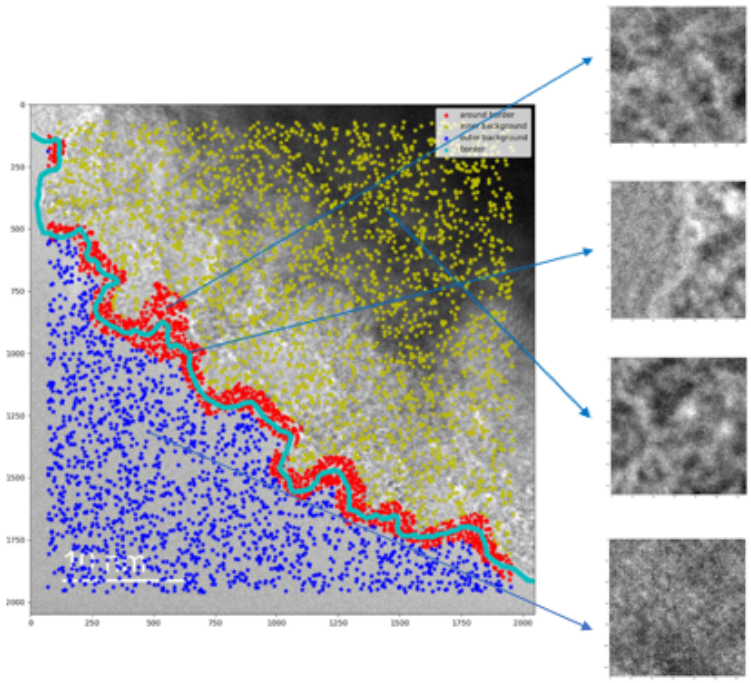

Figure 17: Training data distributions

Before training model, we normalize these sub-images first. As a primary algorithm to speed up convergence of network, normalization turns all value in a small range. For images, the intensities are distributed between 0 and 255, so the easier way to normalize intensity by multiplying values with a scalar. Therefore, we create a normalization function and it is defined as equation3.1. Based on the equation, each pixel will be multiplied by "$\frac{1.0}{255.0}$" to get normalized results.

$$y = x \times \frac{1.0}{255.0} \tag{3.1}$$

3.3 CNN Architecture

To get a precise border prediction, we choose the Resnet[5] framework as our basic model and modify part of structures to adapt to our data, so our model is mainly constructed by residual blocks[5]. In this project, considering the fact that the size of sub-images is 129x129, we remove some pooling layers and residual blocks to decrease the depth of this whole network, which makes the network be compatible with the inputs, and the network architecture is given in the table1. From the table, we know that the network is constructed by six convolution layers, eight residual blocks, and one global average pooling, the last part is a linear layer, the number of the final output is 2.

First, we display the structure of the first special residual block in figure18. In this figure, it's easily found that this block is composed of two branches. In the right branch, 1x1 kernel is applied, which keeps the size of feature maps; in the left branch, inputs are convolved with three kinds of kernel sequentially, and in all kernels, "SAME" padding is applied and the stride is chosen as 1, then output maps have the same shape as inputs. At the end of this block, outputs that are generated by the left and right branches will be combined together and become inputs for the following layers.

As the main components of the network, residual blocks provide an important tool to construct a faster and more stable network. The figure19 displays the structure of general residual blocks, and the residual blocks have two different structures: one has a kernel in the

Table 1: Network architecture

Type	kernel size	strides	rate	output size	ops
convolution	3x3	1		127x127x8	72
convolution	3x3	1		127x127x8	576
residual				127x127x16	768
residual *a*				127x127x16	2816
residual *b*				64x64x32	3072
residual *a*				64x64x32	3328
residual *b*				32x32x64	3840
residual *a*				32x32x64	13312
residual *b*				16x16x64	13312
residual *a*				16x16x64	13312
atrous convolution	3x3		2	12x12x64	36864
atrous convolution	3x3		2	8x8x64	36864
convolution	1x1	1		8x8x64	4096
average pool	8x8	1		1x1x64	
convolution	1x1	1		1x1x64	4096
linear				1x1x2	128

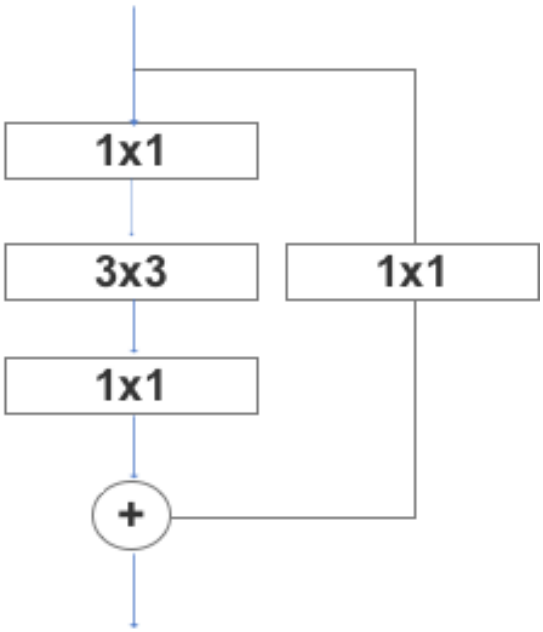

Figure 18: Architecture for the first special residual block

right branch, the other doesn't. In these residual blocks, both left branches are constructed by several kernels, but the tiny difference between these residual blocks. In the residual a blocks, the right branch only takes the inputs and then add them to the outcome from the left branch. While in the left branch, input maps will be convolved with three kernels, which consist of two filters sizes include 1x1 and 3x3. In this residual block, the first and second 1x1 kernel may have a different depth. For all kernels in residual a blocks, the stride is set to 1 and the padding is chosen as "SAME". In the residual b blocks, the left branch has the same shape kernels as block a, while in the right branch, the 1x1 kernel is adopted. Compared with residual a blocks, this residual b blocks still use "SAME" padding but the stride is set as 2. For all residual blocks, the output from the left and right branch will be summed up and become inputs in the next layers. These two kinds of residual blocks have a different stride, as a result, the output size will be different. For the residual a, the size of inputs and output are the same, while the residual b generate output maps with half width and half height of original maps. Based on what we get now, residual blocks takes big advantages over other convolution structure. On the one hand, residual blocks prevent weights from saturating in deep networks. To update weights in the very beginning layers of networks, the gradient must be transferred from the end of networks to where they are, with shallow kernels in the right branch of residual blocks, it's easy to carry gradients when implementing backpropagation. On the other hand, residual blocks decrease the number of parameters required in networks. Taking the second residual a block as an example, the table2 shows the number of parameters needed in a residual block and for a normal convolution structure. In the table, the output maps from the residual block and traditional convolution layer have the same shapes, but the residual block only requires three-fourths of weights that are used in the convolution layer.

Considering that information in long distance will improve the accuracy of detection[21], we add two dilated convolution layers in our network, and the optional parameters "rate" is set to 2. By setting rate as 2, "0" will be filled between rows and columns in kernels, so the actual size of kernel turns to be bigger. The size of expanded kernels is defined by equation3.2, the "o" denotes the new kernels' size, "i" denotes the original kernels' size, and "rate" is an optional parameter.

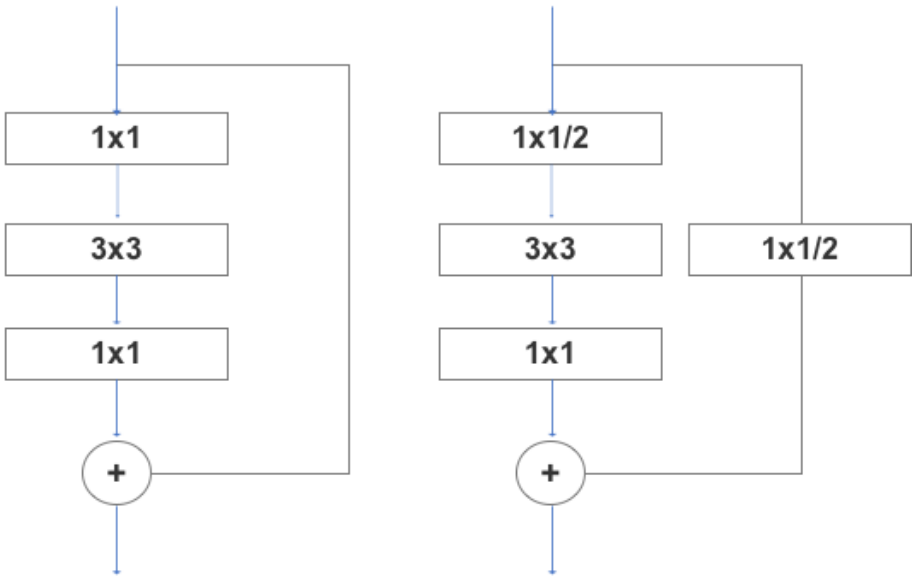

Figure 19: Residual blocks: residual a(left), residual b(right)

Table 2: Number of weights

	residual block	convolution block
input	64x64x32	
filter 1	1x1x16x16	
filter 2	3x3x16x16	3x3x16x32
filter 3	1x1x16x32	
output	64x646x32	
weights	3072	4608

$$o = i + (i - 1) * (rate - 1) \tag{3.2}$$

3.4 Model Training

In machine learning, it takes many hours to train the network, so we run our model on a computer cluster. During training, we allocate many hardware sources and load much software into that cluster. For the hardware, we get one 24-core Xeon Gold 6126(2.60 GHz) CPU, four Titanx GPUs, and 64G memory. And the model is written with python, so we load Tensorflow and Cuda package to run the model, the version of python is 3.6 and the Tensorflow is 1.8.

For the whole training process, the model runs about 30000 iterations, and it takes nearly 10 hours to finish the whole training. mini-batch training method[22] is applied, and the mini-batch size is set to be 64. There are also some other hyper-parameters for the network. In order to increase the convergence speed, a stochastic gradient descent with momentum algorithm[23] is introduced. This method combines the current moving speed and the previous moving speed, such that it will speed up in one direction in which the gradient moves slowly and slow down where the gradient changes a lot. The equation3.3 gives the definition, in the equation, V_t denotes the moving speed at time t, "V_{t-1}" denotes the moving speed at time $t-1$, "$\nabla_w L(W, X, y)$" denotes the derivative of loss with respect to the weight "w", "α" denotes the learning rate. The initial learning rate is chosen and set to be 0.001, and this learning rate will decay by four percent in every 2000 iterations. The momentum is initialized to be 0.99, and it decreases by 10 percent in every 2000 iterations.

$$\begin{aligned} V_t &= \beta * V_{t-1} + (1-\beta) * \nabla_w L(W, X, y) \\ W &= W - \alpha * V_t \end{aligned} \tag{3.3}$$

In deep learning, it's easy for a network to suffer from overfitting problems. To address this issue, we apply a batch normalization method[24] in all convolution layers except the last output layer. The batch normalization method provides us a good way to prevent overfitting problem, and the procedure is given by Algorithm 1. In the batch normalization algorithm, the parameters "γ" and "β" will be learned from the training data. With batch normalization, the result that is generated by small-batch training data will be normalized with zero-center and variance as 1, thus, the covariance in data significantly decreases. After

normalization, values will be shifted to a target position which represents the distribution of whole training data. To implement batch normalization, we set the decay rate to be 0.9 and epsilon to 1e-5.

Algorithm 1 Batch Normalization Algorithm

Input: Values of x over a mini-batch: $B = \{x_1 ... x_m\}$, Parameters to be learned: γ, β

Output: y_i

$\mu_b \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$

$\sigma_b^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_b)^2$

$\hat{x_i} \leftarrow \frac{(x_i - \mu_b)}{\sqrt{\sigma_b^2 + \epsilon}}$

$y_i \leftarrow \gamma \hat{x_i} + \beta$

3.5 Model Measurements

When training model, we have to choose some metrics to test the performance of this network, so some measurements are introduced, including accuracy, precision and sensitivity et al.. In the CNN model, the network will output the predicted label of the given input. Based on these predicted results and objects' actual labels, we will obtain four values, which are true positive, true negative, false positives, and false negative, the table3 describes the definitions. As the table indicates, true positives denotes that data which is labeled as border is predicted as border, false positives show that these that are labeled as background is predicted as border, false negatives indicates that data which is labeled as border is predicted as background, true negatives gives that these that are labeled as background is predicted as background. After getting these values, we can easily derive the results about accuracy, precision, and others. The multiple results largely decrease the bias from only one measurement, especially in unbalanced classification problems. Considering one situation that one class is dominant in all data, even we get the wrong prediction for all the minor class data, the overall accuracy is still pretty high. Therefore, other metrics like precision and sensitivity, specificity are necessary when evaluating the performance of networks. According to the result we get, the total accuracy is calculated by equation3.4,

it indicates the percentage of all correctly predicted data. And we also use the precision to show the percentage of data which is correctly predicted as a border in the whole border region, and the definition is given by equation3.5. And the equation3.6 gives the definition of sensitivity, it indicates the percentage pixels that are correctly predicted as border take in all pixels which are predicted as border, and the specificity is defined by equation3.7, it shows the percentage of correctly predicted pixel as background in all pixels as background, the last one is F1 score, it combines the precision and sensitivity, and the definition is given by equation3.8. Based on these measurements, we can evaluate the model in different perspectives.

Table 3: Confusion matrix

		Actual class	
		border	background
Predicted class	border	True Positives(TP)	False Positives(FP)
	background	False Negatives(FN)	True Negatives(TN)

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.6}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.7}$$

$$F1 = 2 * \frac{precision + sensitivity}{precision * sensitivity} \tag{3.8}$$

4.0 Results

After the model is properly trained, we freeze the parameter and apply this model on the testing set. In the testing phase, we conduct the following procedures: First, to guarantee that segmented images have the same shape with original images, the shape of which is 2048x2048, we implemented "same" padding to the test images, then the size of new padded images is expended to be 2176x2176. Second, we crop these new images into sub-images for each pixel and then those sub-images are fed into the network to get the corresponding predicted label. Finally, when getting the predicted label for every single point, we built the segmented images with those label predictions. The figure20 shows the result generated by machine learning and image processing. In the figure, the middle column of images indicate predicted images from CNN model, and the right column of images represent the segmented images with "canny" edge detection algorithm. In the predicted images, the black area denotes background, and the white area implies the predicted border area. Based on the result, we can get that it's hard for the image processing method to detect the materials' border, especially in second and third images. In the second images generated by edge detection, it marks the projection of materials instead of the border, and this result highly depends on the threshold we adopt when processing images. The image processing method displays a bad performance to detect the materials' border in the third image, no matter what threshold we use, it only outputs the saddle points. However, compared with the images obtained by the edge detection algorithm, the predicted borders in the CNN model have higher accuracy.

Moreover, to check if the predicted border locates at the border of materials, we choose one segmented result and combine it with the real material's border, figure21 shows the combined result. In this figure, the black line in the white area denotes the real border of material. Based on the result, it implies that the model has the ability to predict the border of material, but the predicted border is thicker than the real one. This situation suggests that the model can't tell the difference in sub-images which have adjacent points as a central point.

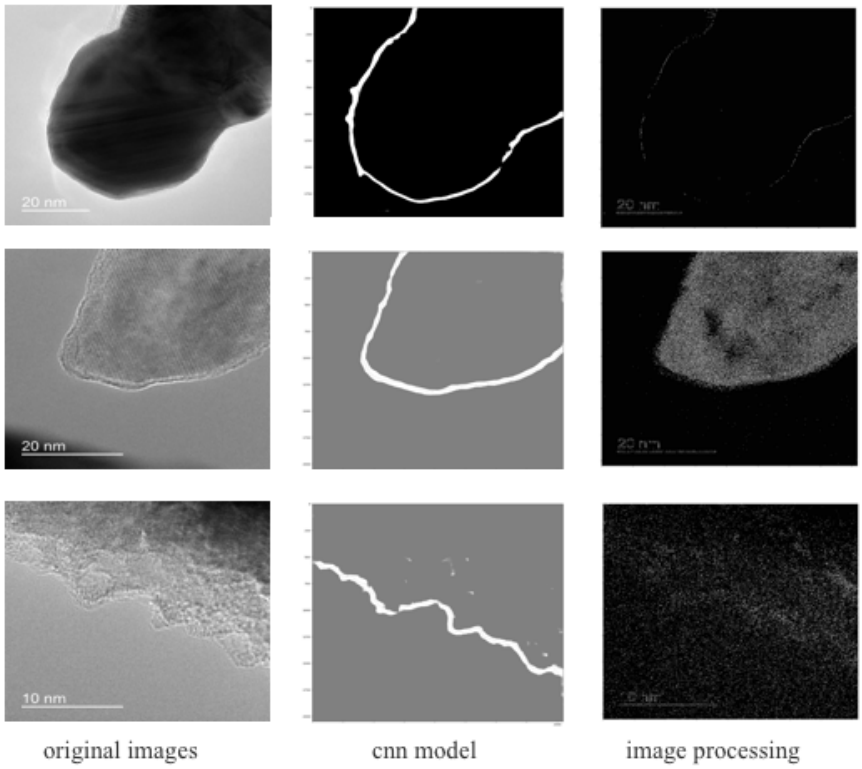

Figure 20: Border detection

Based on the result, the incorrectly predicted border points largely affect the accuracy of this detection task. To reduce the number of incorrectly predicted points and narrow the width of the predicted border, several post-process methods are proposed. One method is that we add a threshold to the predicted probability. By setting a threshold, only if the probability that the model outputs is higher than this threshold point will be assigned to the predicted class. Consequently, the method decreases the uncertainty about predicted labels. To demonstrate the effect from threshold on segmented images, we pick up some thresholds and draw a diagram to display the relationship between measurements with different thresholds. In the figure22, the precision, specificity, F1 score, and overall accuracy increase as threshold increases from 0.5 to 0.95, but sensitivity decreases. When the threshold is 0.5, the specificity and accuracy are nearly 0.96, as threshold increase to 0.95, they approach

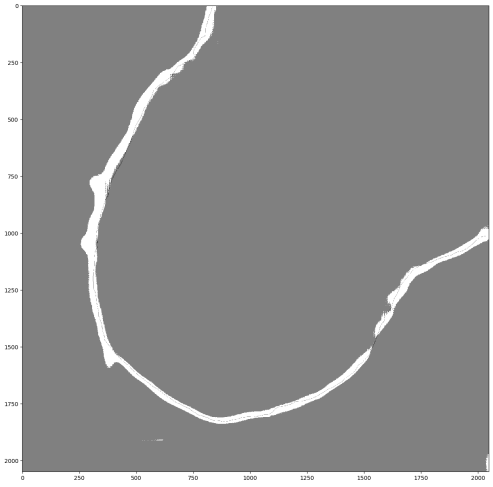

Figure 21: Segmented images and human labeled border

0.995. Both of precision and F1 score are less than 0.08, even the tendency is going up. The sensitivity keeps decreasing from nearly 0.9 to 0.3 when threshold approaches 0.95. The low precision and F1 score indicate that lots of incorrectly predicted border points exist in the segmented images, which is also illustrated by figure23. In this figure, we show the segmented images when thresholds are set as 0.5 and 0.8. In right segmented image with a threshold as 0.8, the border turns to be thinner when compared with left image. Compared with the ground-truth images, the real border is only 1 pixel wide, so even the number of pixels which are predicted as border decreases, there are still tons of incorrectly predicted border points existing in segmented images. Thus, very low precision and F1 score are obtained.

As another traditional image processing algorithm, erosion is chosen to decrease the width of the predicted border. The figure24 shows the relationship between those measurements and kernel size. In the figure, the specificity and accuracy are almost same, and both of them increase from 0.97 to 0.99 as kernel size increases, while the recall keeps decreasing from nearly 0.9 to 0.4. The precision rate and F1 score increase first and then decrease at the same time. The figure25 shows some results we get with the image processing method.

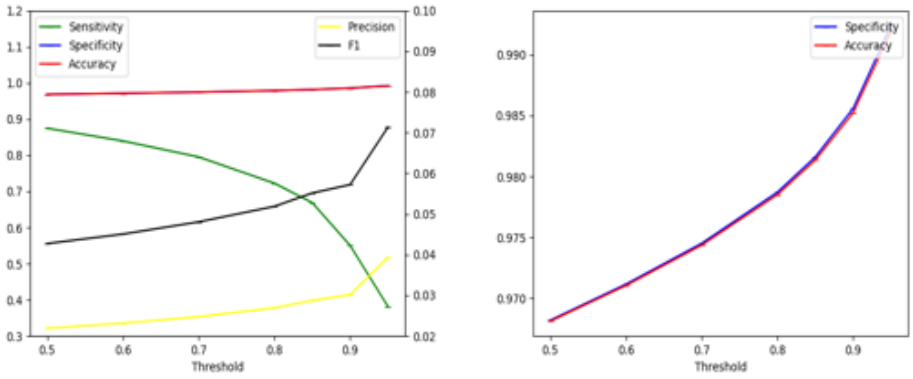

Figure 22: The relationship between measurements and threshold

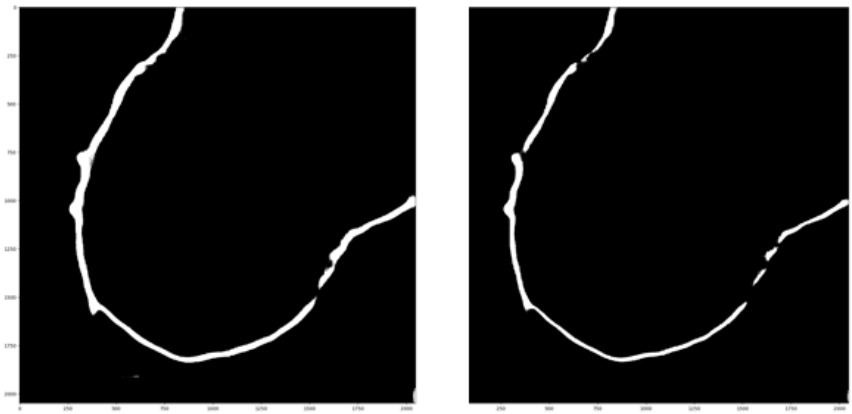

Figure 23: Segmented images: No threshold(left); Threshold as 0.8(right)

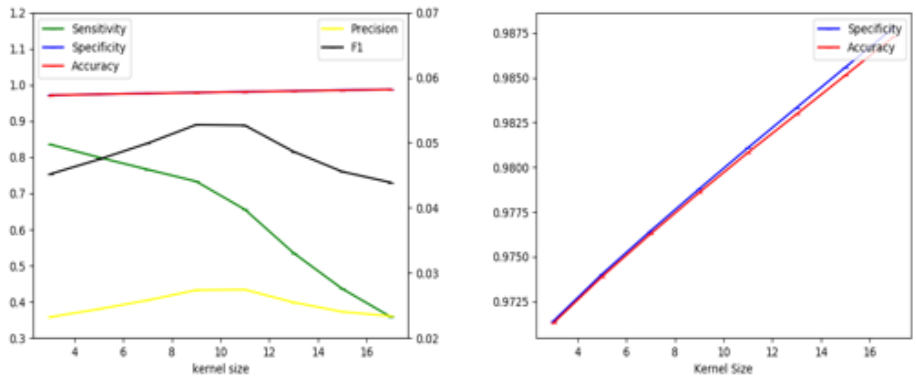

Figure 24: The relationship between measurements and kernel size

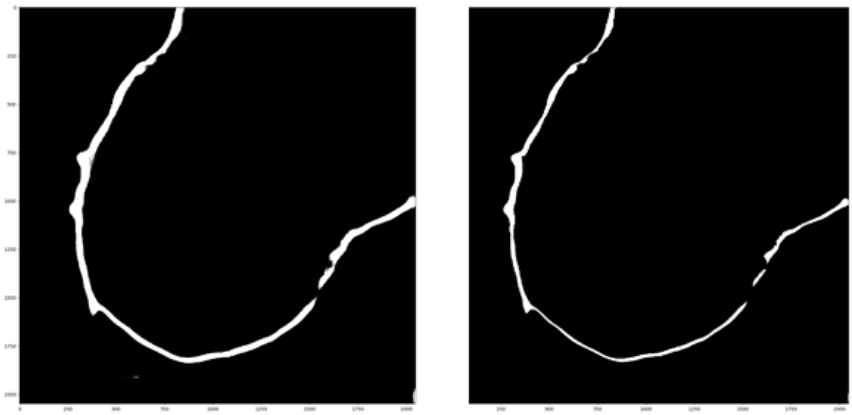

Figure 25: Segmented images with erosion: No erosion; Erosion(11)

In this project, two different post-processing methods are applied to decrease the numbers of an incorrectly predicted border and thus shrink the width of the predicted border. In the segmented images, both of these methods reduce huge incorrectly predicted border points, which is demonstrated by a thinner border when compared to the predicted border without post-processing method. However, Considering the fact that the sensitivity decreases when post-processing method is implemented, we can derive that this post-processing method also reduces the number of points which are correctly predicted as a border, that is a trade-off between the width and the number of correctly predicted border points.

5.0 Future Work

5.1 Possible Development

In the project, the feature detection work is built on small sub-images, and the size of sub-images is fixed and chosen based on observation on sub-images. This choice of size of sub-images is easy to introduce biases to the network. To address this issue, it's worth considering applying several models to process variable sizes of sub-images. Given a central point, several sizes of sub-images are cropped and different probability will be derived, then a signal label is predicted by combining those results from all models. With multiple models and variable sizes of sub-images, the output might be more precise and reliable. We also mention some new architecture for images segmentation above, like fully convolutional networks. Considering the fact that fully convolutional networks output segmented images directly, it could save us lots of time in cropping sub-images and reconstructing images according to labels of pixels in our model. The dilated convolution structure also gives us some inspiration about model architecture, even we add two dilated convolution layers, but we haven't tried to apply multiple rates in one layer. So the idea that choosing variable rate in one layer and combine the output maps might increase the precision about locating border points.

5.2 Current Bottleneck

The current bottleneck of our network is mainly caused by model and data, and several problems are more necessary to be addressed:

(1)The fact that the border is labeled manually by observing the original images with naked eyes causes high bias to models. In general, it's hard to tell which pixel is the real border point in images with a high resolution.

(2)some interpolation method is tried to get a continuous border, which highly affects the quality of training data. Thus, the training data is kind of dirty in training models.

(3)The high computer complexity cost long times in the training model because of the big size of sub-images, even we run it on multiple GPUs with parallel working.

(4)This model is only used for detecting the single border of a single object, so the model won't work to find the border of multiple objects.

6.0 Conclusions

As a powerful tool to capture electrons at a nanoscale, TEM provides us a fundamental method to detect materials' structure and analyze their behaviors. In this project, we propose a CNN model to detect the materials' border in TEM images, which classify the points on the border from that in the background. With this CNN model, we get a pretty higher accuracy to detect the materials' border points, even the predicted border is thicker than the real materials' border. Based on the segmented result, we also implement the post-processing method to narrow the thickness of predicted border and improve the precision for border points, this post-processing method not only reduce the number of incorrectly predicted border points, but also decrease the correctly predicted border, and thus the sensitivity significantly decreases.

To the best of our knowledge, machine learning method starts to be applied in detecting materials in TEM images recently. Previous research only analyzes the application of deep learning in detecting atoms with a modified Unet[25], and this thesis explores the prospect of deep learning method in detecting the border of materials. With deep learning, the border points will be automatically labeled and the performance of detection is relatively good. In this project, the deep learning method also shows its advantage in processing images, especially for those images that traditional image processing method is not able to process, this method still performs well.

Bibliography

[1] LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. doi:10.1145/3065386.

[3] Simonyan, K. & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

[4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. doi:10.1145/3065386.

[6] Cha, K. H., Hadjiiski, L., Samala, R. K., Chan, H. P., Caoili, E. M., & Cohan, R. H. (2016). Urinary bladder segmentation in CT urography using deepâlearning convolutional neural network and level sets. Medical physics, 43(4), 1882-1896.

[7] Sangeethaa, S. N., & Maheswari, P. U. (2018). An Intelligent Model for Blood Vessel Segmentation in Diagnosing DR Using CNN. Journal of medical systems, 42(10), 175.

[8] Khagi, B., & Kwon, G. R. (2018). Pixel-Label-Based Segmentation of Cross-Sectional Brain MRI Using Simplified SegNet Architecture-Based CNN. Journal of Healthcare Engineering, 2018.

[9] Reimer, L. (2013). Transmission electron microscopy: physics of image formation and microanalysis (Vol. 36). Springer.

[10] University of Pittsburgh University Marketing Communications Webteam. (n.d.). Transmission Electron Microscope (TEM). Retrieved from http://www.nano.pitt.edu/node/237

[11] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160, 3-24.

[12] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[14] Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology, 160(1), 106-154.

[15] (n.d.). Retrieved from http://cs231n.github.io/neural-networks-1/

[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

[17] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

[18] Tai, L., Ye, H., Ye, Q., & Liu, M. (2017, July). PCA-aided fully convolutional networks for semantic segmentation of multi-channel fMRI. In 2017 18th International Conference on Advanced Robotics (ICAR) (pp. 124-130). IEEE.

[19] Lane, T., & Lane, T. (2018, November 02). Transposed Convolutions explained with... MS Excel! â Apache MXNet â Medium. Retrieved from https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8.

[20] Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122.

[21] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

[22] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[23] Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. ICML (3), 28(1139-1147), 5.

[24] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[25] Madsen, J., Liu, P., Kling, J., Wagner, J. B., Hansen, T. W., Winther, O., & SchiÃ¸tz, J. (2018). A Deep Learning Approach to Identify Local Structures in AtomicâResolution Transmission Electron Microscopy Images. Advanced Theory and Simulations, 1(8).